Name : Syed Husban Rahat

ID:- 19030302000 2

Term Test : 2

Course Title : Computer Architecture

Course Code : CSE-311

Answer to the question: Q1

(a)

There are several drawbacks of a Single-cycle Implementation. For those drawbacks a Single-cycle Implementation is not used today.
Drawbacks of a Single-cycle Implementation are given below:—

- Hardware units can only be used once in the cycle.
  1) Some must be replicated (ALU, memory)
  2) Increased hardware costs.
- All instructions must complete in 1 cycle (CPI=1)
  1) Clock cycle set to the longest instruction.
  2) Different instructions do different amounts of work,
     for example:
     - add uses instruction memory, register file, ALU, register file again.
     - lw uses instruction memory, register file, ALU, data memory, register file again.

Advantages of pipelined Datapath :-

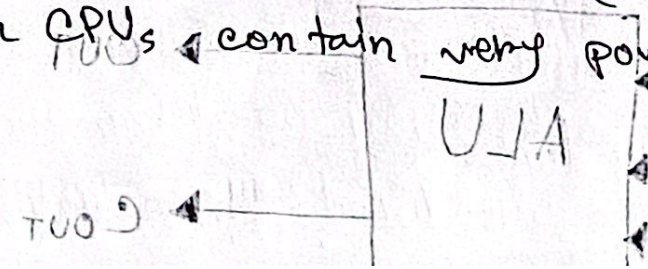
1) Instruction throughput increases.
2) Increase in the number of pipeline stages increases the number of instructions executed simultaneously.
3) Faster ALU can be designed when pipelining is used.
4) Pipelined CPU's works at higher clock frequencies than the RAM.
5) Pipelining increases the overall performance of the CPU.

(b)

ALU:

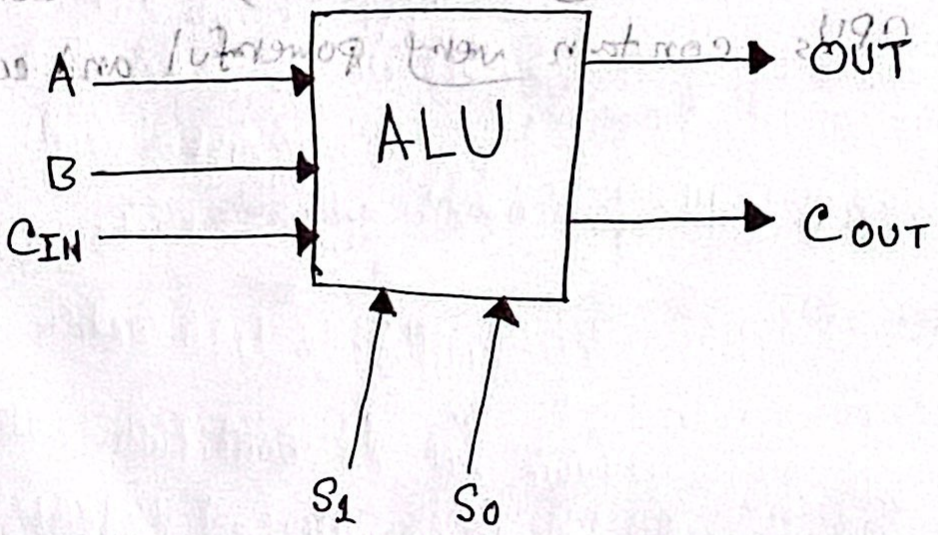
ALU is the short form of Arithmetic Logic Unit.

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs.

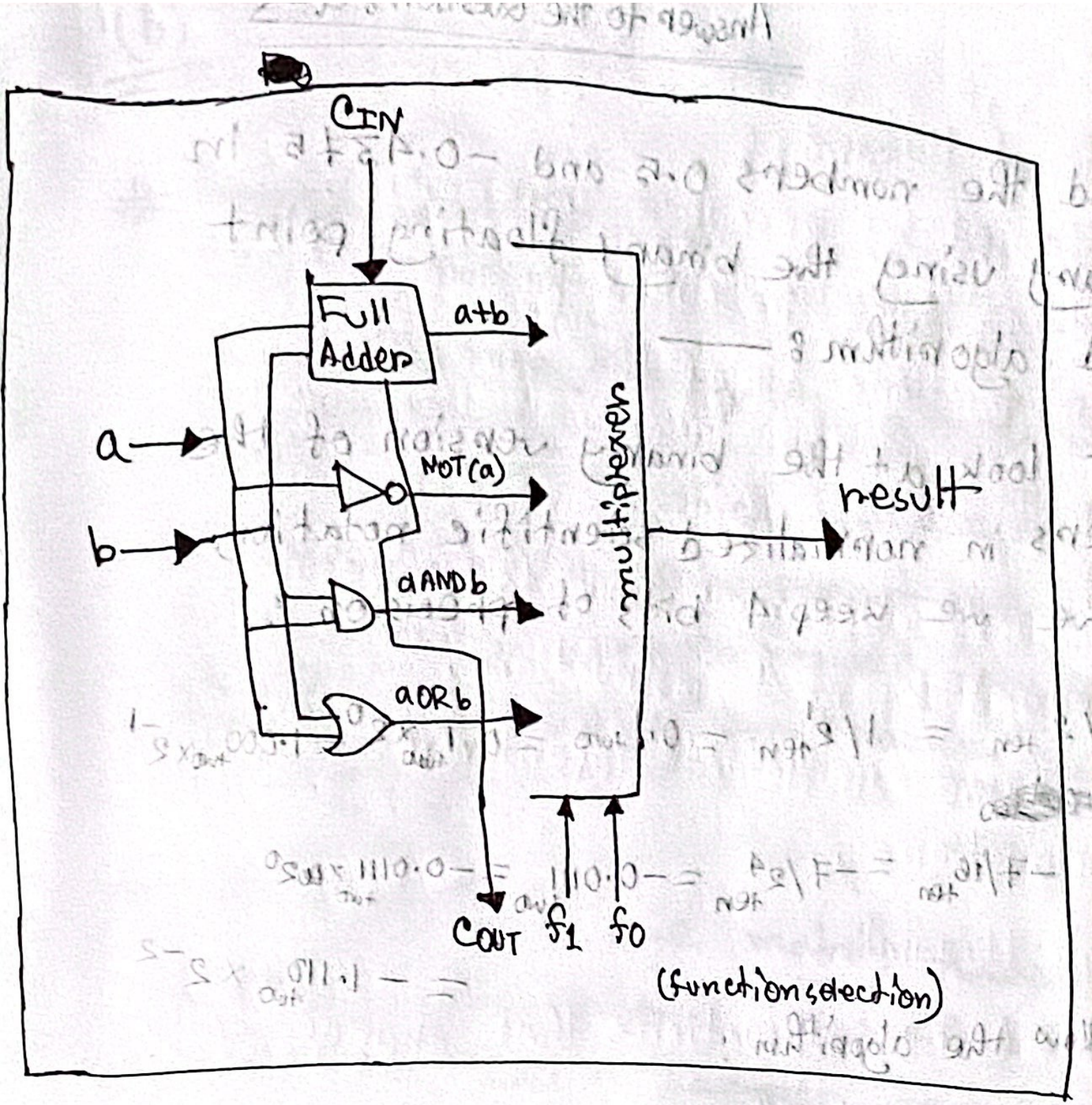


Design a 1-bit ALU:

Below is a block diagram of a simplified ALU. Let's give it a carry-in and a carry-out, and let that output bit be low in any case where a carry is not generated. Two select lines should determine which operation the ALU performs from the set {AND, OR, XOR, ADD}



All the components that you need to use are already in the simulator. You will need the basic gates, adder, multiplexer to complete the circuit. Build the circuit in steps and make sure to go through the tutorial about Logism to understand the wiring.



(a)

Answer to the Question : Q2

Add the numbers 0.5 and −0.4375 in binary using the binary floating point add algorithm :—

⇒ Let's first look at the binary version of the two numbers in normalized scientific notation, assuming we keep 4 bits of precision :

$0.5_{ten} = 1/2_{ten} = 1/2^1_{ten} = 0.1_{two} = 0.1_{two} \times 2^0 = 1.000_{two} \times 2^{-1}$

$-0.4375_{ten} = -7/16_{ten} = -7/2^4_{ten} = -0.0111_{two} = -0.0111_{two} \times 2^0$

$= -1.110_{two} \times 2^{-2}$

Now, we follow the algorithm :

Step: 1 The significand of the number with the lesser exponent ($-1.11_{two} \times 2^{-2}$) is shifted right until its exponent matches the larger number :

$-1.110_{two} \times 2^{-2} = -0.111_{two} \times 2^{-1}$

Step: 2 Add the significands :

$1.000_{two} \times 2^{-1} + (-0.111_{two} \times 2^{-1})$

$= 0.001_{two} \times 2^{-1}$

Step: 3

Normalize the sum, checking for overflow or underflow:

$0.001_{two} \times 2^{-1} = 0.010_{two} \times 2^{-2} = 0.100_{two} \times 2^{-3}$

$= 1.000_{two} \times 2^{-4}$

There is no overflow or underflow.

Step: 4

Round the sum:

$1.000_{two} \times 2^{-4}$

The sum already fits exactly in 4 bits, so there is no change to the bits due to rounding!

This sum is then

$1.000_{two} \times 2^{-4} = 0.0001000_{two}$

$= 0.0001_{two}$

$= 1/2^4_{ten}$

$= 1/16_{ten}$

$= 0.0625_{ten}$

∴ This sum is what we would expect from this addition of 0.5_{ten} and -0.4375_{ten}. As.

(b)

Program Counter:

A program counter (PC) is a CPU register in the computer processor which has the address of the next instruction to be executed from memory. It is a digital counter needed for faster execution of tasks as well as for tracking the current execution point.

Actually there are mainly three types of hazards are available in pipeline hazards. Types of different hazards are given below:-

1) Structural Hazards.
2) Data Hazards.
3) Control Hazards.

(d)

(1) Structural Hazards:

- Caused by resource contention
- Using same resource by two instructions during the same cycle.

(2) Data Hazards:

- An instruction may compute a result needed by next instruction.
- Hardware can detect dependencies between instructions.

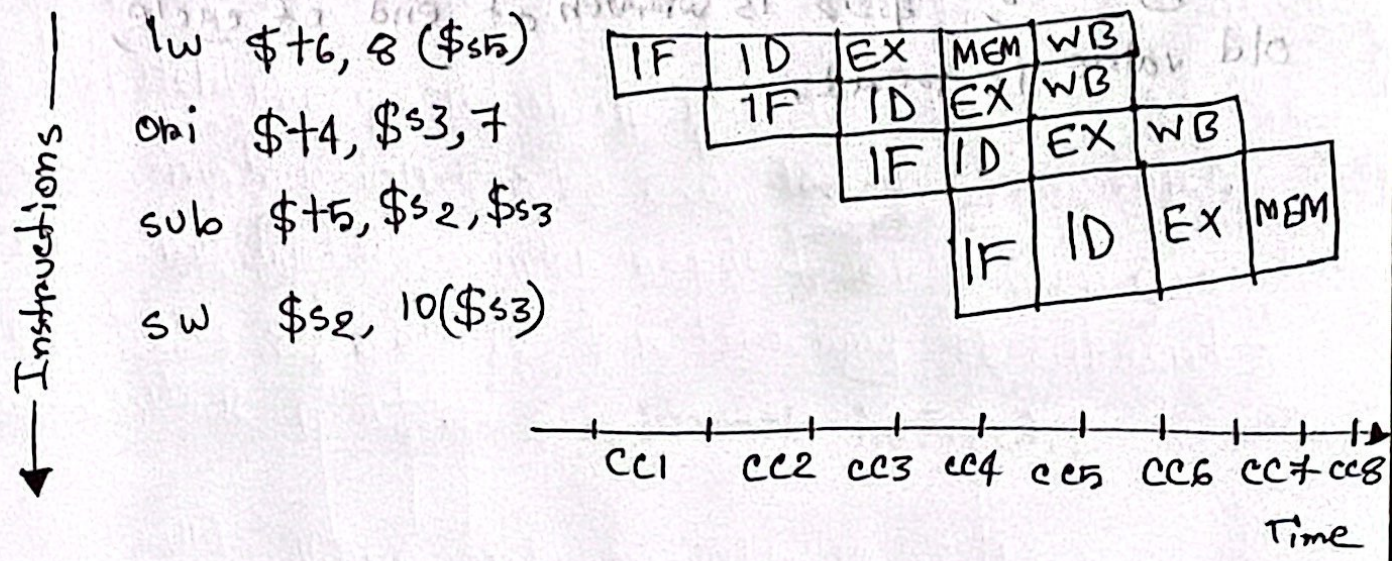
(3) Control Hazards:

- Caused by instructions that change control flow. (branches/jumps)
- Delays in changing the flow of control.

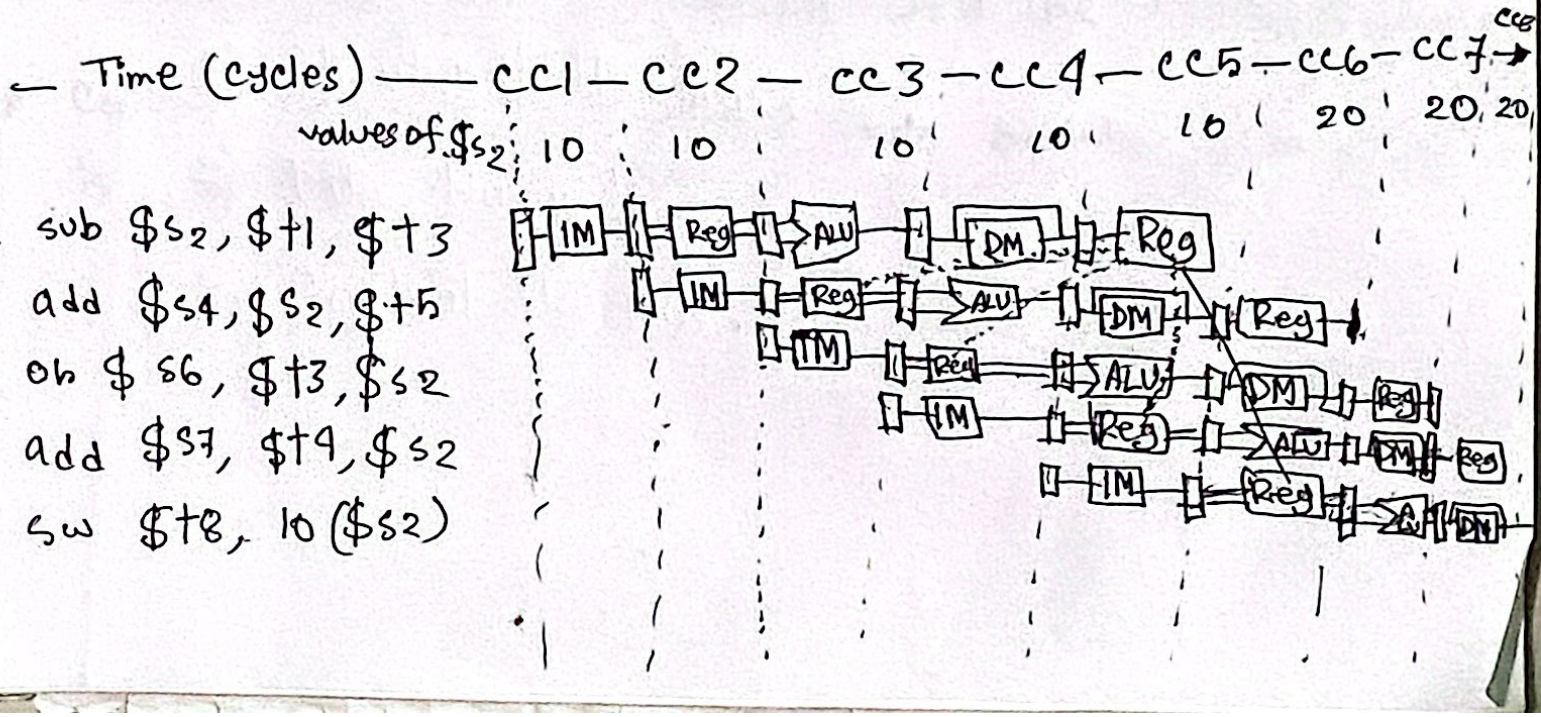
Examples:

Example of Structural Hazards:

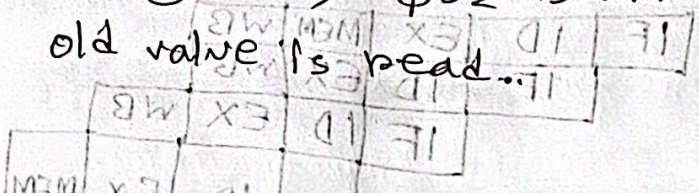
- Writing back ALU result in stage 4.
- Conflict with writing load data in stage 5.



Example of Data Hazards:



- Result of sub is needed by add, or, and sw instructions
- Instructions add, $ or will read old value of s_2 from reg file.
- During cc5, s_2 is written at end of cycle, old value is read.

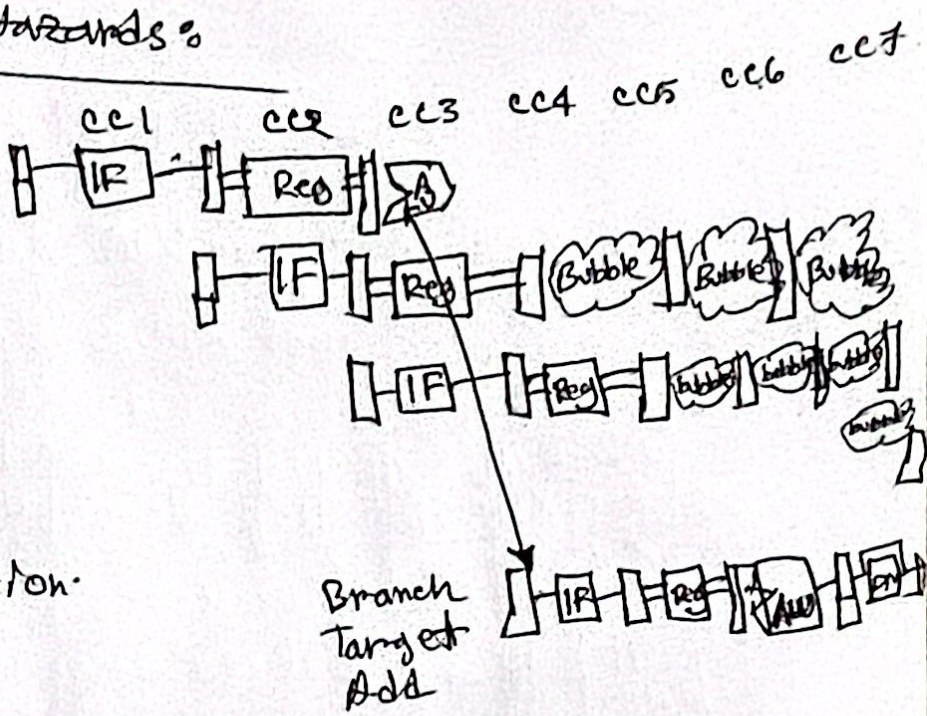


Example of control Hazards:

Beq t_1, t_2, L1

Next 1

Next 2



L1 : target instruction.

- Control logic detects a Branch instruction in the 2nd stage.
- ALU computes the Branch outcome in the 3rd stage.
- Next 1 and Next 2 instruction will be fetched anyway.
- Convert Next 1 and Next 2 into bubbles if branch is taken.